Introduction to Modelica modeling and the OpenModelica and MathModelica tools

Invited talk to workshop
"Can Systems biology aid personalized medication?"

December 5, 2011

Peter Fritzson

Professor at Linköping University, Sweden
Vice Chairman of Modelica Association
Director of Open Source Modelica Consortium
peter.fritzson@liu.se
Introduction to Modelica
Model knowledge is stored in books and human minds which computers cannot access

“The change of motion is proportional to the motive force impressed“

– Newton
Modelica Background: The Form – Equations

- Equations were used in the third millennium B.C.
- Equality sign was introduced by Robert Recorde in 1557

\[ v = \frac{\text{INTEG}(F)}{m} \]

Newton still wrote text (Principia, vol. 1, 1686)
“The change of motion is proportional to the motive force impressed”

CSSL (1967) introduced a special form of “equation”:
\[ \text{variable} = \text{expression} \]

Programming languages usually do not allow equations!
What is Modelica?

A language for modeling of complex cyber-physical systems

- Robotics
- Control
- Automotive
- Aircraft
- Satellites
- Power plants
- Systems biology
What is Modelica?

A language for modeling of complex cyber physical systems

i.e., Modelica is not a tool

Free, open language specification:

There exist several free and commercial tools, for example:

• OpenModelica from OSMC
• MathModelica from MathCore
• Dymola from Dassault systems
• SimulationX from ITI
• MapleSim from MapleSoft

Available at: www.modelica.org
Modelica – The Next Generation Modeling Language

Declarative language
Equations and mathematical functions allow acausal modeling, high level specification, increased correctness

Multi-domain modeling
Combine electrical, mechanical, thermodynamic, hydraulic, biological, control, event, real-time, etc...

Everything is a class
Strongly typed object-oriented language with a general class concept, Java & MATLAB-like syntax

Visual component programming
Hierarchical system architecture capabilities

Efficient, non-proprietary
Efficiency comparable to C; advanced equation compilation, e.g. 300 000 equations, ~150 000 lines on standard PC
What is *acausal* modeling/design?

Why does it increase *reuse*?

The acausality makes Modelica library classes *more reusable* than traditional classes containing assignment statements where the input-output causality is fixed.

Example: a resistor *equation*:

\[ R \cdot i = v; \]

can be used in three ways:

\[ i := v/R; \]
\[ v := R \cdot i; \]
\[ R := v/i; \]
What is Special about Modelica?

Multi-Domain Modeling

Cyber-Physical Modeling

Physical
- electric
- mechanics
- control

Cyber

Reference

Electric

R L

emf

Axis

Bearing

Angle-Sensor

PID Control System
What is Special about Modelica?

- **Multi-Domain Modeling**
  - Keeps the physical structure

- **Acausal model** (Modelica)

- **Causal block-based model** (Simulink)

- **Visual Acausal Hierarchical Component Modeling**
What is Special about Modelica?

- **Hierarchical system modeling**
- **Visual Acausal Hierarchical Component Modeling**

```
Srel = n*transpose(n)+(identity(3)-n*transpose(n))\cdot \cos(q)-
\text{skew}(n)\cdot \sin(q);

wrela = n*qd;
zrela = n*qdd;

Eb = Sa*transpose(Srel);

r0b = r0a;
vb = Srel*va;
wb = Srel*(wa + wrela);
ab = Srel*aa;
zb = Srel*(za + zrela + cross(wa, wrela));
```
What is Special about Modelica?

Multi-Domain Modeling

A **textual class-based language**

Object-Oriented mainly used as structuring concept

**Behaviour described declaratively using**

- Differential algebraic equations (DAE) (continuous-time)
- Event triggers (discrete-time)

```modelica
class VanDerPol  "Van der Pol oscillator model"
  Real x(start = 1)  "Descriptive string for x";
  Real y(start = 1)  "y coordinate";
  parameter Real lambda = 0.3;

  equation
    der(x) = y;
    der(y) = -x + lambda*(1 - x*x)*y;

end VanDerPol;
```

Visual Acausal Hierarchical Component Modeling

Typed Declarative Equation-based Textual Language

Variable declarations

Differential equations
What is Special about Modelica?

- **Multi-Domain Modeling**
- **Visual Acausal Component Modeling**
- **Typed Declarative Equation-based Textual Language**

**Hybrid modeling** = continuous-time + discrete-time modeling
Graphical Modeling - Using Drag and Drop Composition
A DC motor can be thought of as an electrical circuit which also contains an electromechanical component.

```model DCMotor
  Resistor R(R=100);
  Inductor L(L=100);
  VsourceDC DC(f=10);
  Ground G;
  ElectroMechanicalElement EM(k=10,J=10, b=2);
  Inertia load;

  equation
    connect(DC.p, R.n);
    connect(R.p, L.n);
    connect(L.p, EM.n);
    connect(EM.p, DC.n);
    connect(DC.n, G.p);
    connect(EM.flange, load.flange);

end DCMotor
```
Corresponding DCMotor Model Equations

The following equations are automatically derived from the Modelica model:

<table>
<thead>
<tr>
<th>Equation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 == DC.p.i + R.n.i</td>
<td>EM.u == EM.p.v - EM.n.v</td>
</tr>
<tr>
<td>DC.p.v == R.n.v</td>
<td>0 == EM.p.i + EM.n.i</td>
</tr>
<tr>
<td>0 == R.p.i + L.n.i</td>
<td>EM.i == EM.p.i</td>
</tr>
<tr>
<td>R.p.v == L.n.v</td>
<td>EM.u == EM.k * EM.ω</td>
</tr>
<tr>
<td>0 == L.p.i + EM.n.i</td>
<td>R.u == R.R * R.i</td>
</tr>
<tr>
<td>L.p.v == EM.n.v</td>
<td>EM.J * EM.ω == EM.M - EM.b * EM.ω</td>
</tr>
<tr>
<td>0 == DC.p.i + DC.n.i</td>
<td>L.u == L.p.v - L.n.v</td>
</tr>
<tr>
<td>0 == EM.p.i + DC.n.i</td>
<td>0 == L.p.i + L.n.i</td>
</tr>
<tr>
<td>EM.p.v == DC.n.v</td>
<td>DC.u == DC.p.i</td>
</tr>
<tr>
<td>0 == DC.n.i + G.p.i</td>
<td>0 == DC.p.i</td>
</tr>
<tr>
<td>DC.n.v == G.p.v</td>
<td>EM.p.v == DC.n.v</td>
</tr>
</tbody>
</table>

Automatic transformation to ODE or DAE for simulation:

\[
\frac{dx}{dt} = f[x, u, t] \quad \text{and} \quad g\left[\frac{dx}{dt}, x, u, t\right] = 0
\]
Model Translation Process to Hybrid DAE to Code

Modeling Environment

Modelica Graphical Editor
Modelica Textual Editor
Modelica Model

Modelica Model

Modelica Source code
Translator
Analyzer
Optimizer
Code generator
C Compiler
Simulation

Frontend

“Middle-end”

Backend

Flat model Hybrid DAE
Sorted equations
Optimized sorted equations
C Code
Executable
Modelica in Power Generation
GTX Gas Turbine Power Cutoff Mechanism

Developed by MathCore for Siemens

Courtesy of Siemens Industrial Turbomachinery AB, Finspång, Sweden
Application of Modelica in Robotics Models
Real-time Training Simulator for Flight, Driving

• Using Modelica models generating real-time code
• Different simulation environments (e.g. Flight, Car Driving, Helicopter)
• Developed at DLR Munich, Germany
• Dymola Modelica tool

Courtesy of Martin Otter, DLR, Oberphaffenhofen, Germany
Modelica Examples – Systems Biology

Cardiovascular System:
Francois Cellier et Al, ETH Zürich
Modelica Examples – Systems Biology

HumMod
Kofranek et Al., Charles University, Prag
Cell Biology Insulin Receptor Model (Sedgehat et al)
Modelica Modeling Using a free PathWay Library (BioChem)
The BioChem Library for PathWay Modeling

- Free Open Source Library
- Originally Developed at PELAB/IDA LIU 2003-2006, continued development at MathCore Engineering AB and LIU
- Also used for SBML to Modelica mapping

Several BioChem Slides, Courtesy of Jan Brugård, MathCore Engineering AB/ Wolfram Research
model EnzMM "An enzymatic reaction with Michaelis-Menten kinetics"
  extends BioChem.Compartments.Compartment;
  BioChem.Substances.Substance F6P(c.start=2) "Fructose-6-phosphate";
  BioChem.Reactions.MichaelisMenten.Uur uur(vF=1.5, KmS=0.1, KmP=0.05);
  BioChem.Substances.Substance G6P(c.start=1) "Glucose-6-phosphate";
  equation
    connect(G6P.n1,uur.s1);
    connect(uur.p1,F6P.n1);
  end EnzMM;
The Modelica Standard Library contains components from various application areas, including the following sublibraries:

- **Blocks**: Library for basic input/output control blocks
- **Constants**: Mathematical constants and constants of nature
- **Electrical**: Library for electrical models
- **Icons**: Icon definitions
- **Fluid**: 1-dim Flow in networks of vessels, pipes, fluid machines, valves, etc.
- **Math**: Mathematical functions
- **Magnetic**: Magnetic Fluxtubes – for magnetic applications
- **Mechanics**: Library for mechanical systems
- **Media**: Media models for liquids and gases
- **SIunits**: Type definitions based on SI units according to ISO 31-1992
- **Stategraph**: Hierarchical state machines (analogous to Statecharts)
- **Thermal**: Components for thermal systems
- **Utilities**: Utility functions especially for scripting
Brief Modelica History

• First Modelica design group meeting in fall 1996
  • International group of people with expert knowledge in both language design and physical modeling
  • Industry and academia

• Modelica Versions
  • 1.0 released September 1997
  • 2.0 released March 2002
  • 2.2 released March 2005
  • 3.0 released September 2007
  • 3.1 released May 2009
  • 3.2 released May 2010
  • 3.3 planned Spring 2012

• Modelica Association established 2000
  • Open, non-profit organization
Modelica Conferences

• The 1\textsuperscript{st} International Modelica conference October, 2000
• The 2\textsuperscript{nd} International Modelica conference March 18-19, 2002
• The 3\textsuperscript{rd} International Modelica conference November 5-6, 2003 in Linköping, Sweden
• The 4\textsuperscript{th} International Modelica conference March 6-7, 2005 in Hamburg, Germany
• The 5\textsuperscript{th} International Modelica conference September 4-5, 2006 in Vienna, Austria
• The 6\textsuperscript{th} International Modelica conference March 3-4, 2008 in Bielefeld, Germany
• The 7\textsuperscript{th} International Modelica conference Sept 21-22, 2009, Como, Italy
• The 8\textsuperscript{th} International Modelica conference March 20-22, 2011 in Dresden, Germany
• The 4\textsuperscript{th} Int. OpenModelica Workshop, Febr 6, 2012, Linköping, Sweden
Modelica Environments and OpenModelica
MathModelica – MathCore / Wolfram Research

- Wolfram Research
- USA, Sweden
- General purpose
- Mathematica integration
- www.wolfram.com
- www.mathcore.com

Car model graphical view

Mathematica
Simulation and analysis
MathModelica – Car Model Simulation & Animation

Courtesy Wolfram Research
PathWay Model Using BioChem and MathModelica
Introductory SBML to Modelica Example

model EnzMM "An enzymatic reaction with Michaelis-Menten kinetics" extends BioChem.Compartments.Compartment; BioChem.Substances.Substance F6P(c.start=2) "Fructose-6-phosphate"; BioChem.Substances.Substance G6P(c.start=1) "Glucose-6-phosphate"; equation connect(G6P.n1,uur.s1); connect(uur.p1,F6P.n1); end EnzMM;
SBML/Modelica Translator

• As the **standard** modeling language within **systems biology** applications **SBML**
  • has a wide range of ready-made models available.
  • a large user base with knowledge about the language and its applications

• By creating a translator to **Modelica** we can give the users access to a **much richer language**, offering new possibilities.
4.2 Model

The definition of Model is shown in Figure 10 on the following page. Only one instance of a Model object is allowed per instance of an SBML Level 2 Version 3 Release 2 document or data stream, and it must be located inside the <sbml> ... </sbml> element as described in Section 4.1. The Model object has an optional attribute id, used to give the model an identifier. The identifier must be a string conforming to the syntax permitted by the XML data type described in Section 5.1.3. Model also has an optional xmlns attribute, of type string. The id and xmlns attributes must be as described in Section 3.3.

Model serves as a container for components of classes FunctionDefinition, UnitDefinition, CompartimentType, SpeciesType, Compartiment, Species, Parameter, InitialAssignment, Rule, Constraint, Reaction and Event. Additional of the classes are placed inside namespaces of classes UnitFunctionDefinitions, UnitDefinitions, InitialAssignments, InitialAssignments, SpeciesDefinitions, ConstraintDefinitions, and ReactionDefinitions. The “root” classes are defined in Figure 10. All of the elements are optional, but if a given bucket container is present within the model, the list must not be empty: that is, it must have length one or more. The resulting XML data object for a full model containing every possible element would have the following form:

```xml
<model id="<id>">
  <function-definition/>
  <unit-definition/>
  <compartiment-type/>
  <species-type/>
  <compartiment/>
  <species/>
  <parameter/>
  <initial-assignment/>
  <reaction/>
  <event/>
  ...</model>
```

Although all the lists are optional, there are dependencies between SBML components such that defining some components requires defining others. An example is that defining a species requires defining a compartment, and defining a reaction requires defining a species. The dependencies are explained throughout the text.

4.5 Class Declarations

Essentially everything in Modelica is a class, from the predefined classes: Real-type and real, to large packages such as the Modelica standard library.

[Example: A rather typical structure of a Modelica class is shown below. A class with a name, containing a number of declarations, followed by a number of equations in an equation section.

```model ClassExample
  parameter Real level[n];
  equation
    component : "Connection to conditional component";
  end ClassExample;
```

The following is the formal syntax of class definitions, including the special variants described in the text:

```model class_name()
  | class: model | extended | block | ( expanding ) connector: type |
  | class: package |
  | class: variable
  | class: operation
  | class: field
  | class: description
  | class: expression
  | class: expression
  | class: expression
  | class: expression
  | class: expression
end class_name;
```

...
Using the BioChem Library and MathModelica for Translation

- Finding a mapping between SBML and the more expressive and general Modelica language
- “Catch” Modelica constructs and map them
- Restrict how the Modelica models are built
  - BioChem library
  - Wizards introduced
Verification of the SBML-Modelica Translator
Part of MathModelica

Models from BioModels database have been used to verify if models that are imported to MathModelica and exported from give the same result as the simulation published on the database.

<table>
<thead>
<tr>
<th></th>
<th>Test models</th>
<th>Succeeded</th>
</tr>
</thead>
<tbody>
<tr>
<td>Import</td>
<td>216</td>
<td>212 (98%)</td>
</tr>
<tr>
<td>Simulation</td>
<td>212</td>
<td>208 (98%)</td>
</tr>
<tr>
<td>Export</td>
<td>18</td>
<td>18 (100%)</td>
</tr>
</tbody>
</table>
Comparison With Other Tools

Models from BioModels database have been used to verify if different tools give the same result as the simulation published on the database.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Test models</th>
</tr>
</thead>
<tbody>
<tr>
<td>MathModelica</td>
<td>98% (42 of 43)</td>
</tr>
</tbody>
</table>
OpenModelica, www.openmodelica.org
The Most Complete Open Source Modelica Tool

- OpenModelica
- Open Source Modelica Consortium (OSMC)
- International
- Open source
- www.openmodelica.org

- OMEdit, graphical editor
- OMOptim, optimization subsystem
OpenModelica (cont.)

- Advanced Interactive Modelica compiler (OMC)
  - Supports most of the Modelica Language
- Basic environment for creating models
  - **OMShell** – an interactive command handler
  - **OMNotebook** – a literate programming notebook
  - **MDT** – an advanced textual environment in Eclipse

- **ModelicaML** – UML Profile
- **MetaModelica** – symbolic manipulation
OSMC – Open Source Modelica Consortium
38 organizational members November 2011

Founded Dec 4, 2007

Open-source community services:
• Website and Support Forum
• Version-controlled source base
• Bug database
• Development courses
• www.openmodelica.org

Code Statistics
# OSMC 38 Organizational Members, Nov 2011

*Initially 7 members, 2007*

<table>
<thead>
<tr>
<th>Companies and Institutes (17 members)</th>
<th>Universities (17 members)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• ABB Corporate Research, Sweden</td>
<td>• Linköping University, Sweden</td>
</tr>
<tr>
<td>• Bosch Rexroth AG, Germany</td>
<td>• Hamburg University of Technology/TuTech, Germany</td>
</tr>
<tr>
<td>• Siemens Turbo Machinery AB, Sweden</td>
<td>• Technical University of Berlin, Germany</td>
</tr>
<tr>
<td>• CDAC Centre for Advanced Computing, Kerala, India</td>
<td>• FH Bielefeld, Bielefeld, Germany</td>
</tr>
<tr>
<td>• CEIT Institute, Spain</td>
<td>• Technical University of Braunschweig, Institute of Thermodynamics, Germany</td>
</tr>
<tr>
<td>• Creative Connections, Prague, Czech Republic</td>
<td>• Technical University of Dortmund, Process Dynamics Group, Germany</td>
</tr>
<tr>
<td>• Fraunhofer FIRST, Berlin, Germany</td>
<td>• Université Laval, modelEAU, Canada</td>
</tr>
<tr>
<td>• Frontway AB, Sweden</td>
<td>• University of Maryland, USA</td>
</tr>
<tr>
<td>• Equa Simulation AB, Sweden</td>
<td>• Georgia Tech, Atlanta, USA</td>
</tr>
<tr>
<td>• Evonik Energy Services, Dehli, India</td>
<td>• Griffith University, Australia</td>
</tr>
<tr>
<td>• IFP, Paris, France</td>
<td>• Politecnico di Milano, Italy</td>
</tr>
<tr>
<td>• InterCAX, Atlanta, USA</td>
<td>• Mälardalen University, Sweden</td>
</tr>
<tr>
<td>• Wolfram/ MathCore USA, Sweden</td>
<td>• Technical University Dresden, Germany</td>
</tr>
<tr>
<td>• Maplesoft, Canada</td>
<td>• Telemark University College, Norway</td>
</tr>
<tr>
<td>• TLK Thermo, Germany</td>
<td>• Ghent University, Belgium</td>
</tr>
<tr>
<td>• VI-grade, Italy</td>
<td>• Ecoles des Mines, CEP, Paris, France</td>
</tr>
<tr>
<td>• VTT, Finland</td>
<td>• University of Ljubljana, Slovenia</td>
</tr>
<tr>
<td>• XRG Simulation, Germany</td>
<td></td>
</tr>
</tbody>
</table>
1 Kalman Filter

Often we don’t have access to the internal states of a system, so we have to reconstruct the state of the system based on the output. The idea with an observer is that we feedback the observed output and adjust the estimation if the estimation is correct then the difference should be small.

Another difficulty is that the measured quantities of a system may be corrupted by disturbances and noise, so the observed output

\[
\begin{bmatrix}
\dot{x} \\
\end{bmatrix} =
\begin{bmatrix}
A & B(u) \\
C & D \\
\end{bmatrix}
\begin{bmatrix}
x \\
u \\
\end{bmatrix} +
\begin{bmatrix}
K & 0 \\
0 & 0 \\
\end{bmatrix}
\begin{bmatrix}
y - \hat{y} \\
0 \\
\end{bmatrix}
\]

Here are $e$ denoting a disturbance in the input signal which can be evaluated by the difference

\[
\begin{align*}
K(y(t))
\end{align*}
\]

By using this quantity as feedback we obtain the observer

\[
\dot{x} = A\hat{x} + Bu
\]

Now form the error as

\[
\begin{align*}
\epsilon &= y - \hat{y} \\
\end{align*}
\]

The differential error is
OpenModelica Demo
OMOptim – Optimization (1)

Model structure

Model Variables

Optimized parameters

Optimized Objectives

![Image of OMOptim interface showing model structure, model variables, optimized parameters, and optimized objectives.](image-url)
Problems
Solved problems
Result plot
Export result data .csv

OMOptim – Optimization (2)

MinEIT

Pareto Front
Modelica Language Interoperability
External Functions – C, FORTRAN 77

It is possible to call functions defined outside the Modelica language, implemented in C or FORTRAN 77

```modelica
function polynomialMultiply
  input Real a[:], b[:];
  output Real c[:];
  output Real c[:]:=zeros(size(a,1)+size(b,1)-1);
end polynomialMultiply;
```

The body of an external function is marked with the keyword `external`.

If no language is specified, the implementation language for the external function is assumed to be C. The external function `polynomialMultiply` can also be specified, e.g. via a mapping to a FORTRAN 77 function:

```modelica
function polynomialMultiply
  input Real a[:], b[:];
  output Real c[:]:=zeros(size(a,1)+size(b,1)-1);
end polynomialMultiply;
```

```fortran
function polynomialMultiply
  input Real a[:], b[:];
  output Real c[:]:=zeros(size(a,1)+size(b,1)-1);
end polynomialMultiply;
```
General Tool Interoperability & Model Exchange
Functional Mock-up Interface (FMI)

The FMI development is part of the MODELISAR 29-partner project
• FMI development initiated by Daimler
• Improved Software/Model/Hardware-in-the-Loop Simulation, of physical models and of AUTOSAR controller models from different vendors for automotive applications with different levels of detail.
• Open Standard
• 14 automotive use cases for evaluation
• > 10 tool vendors are supporting it

Engine with ECU
Gearbox with ECU
Thermal systems
Automated cargo door
Chassis components, roadway, ECU (e.g. ESP)

functional mockup interface for model exchange and tool coupling

courtesy Daimler
OpenModelica FMI Export and Import

- Export: `translateModel FMU(A)`
- `importFMU("A.fmu")`
Faster Simulation – Compiling Modelica to Multi-Core

• **Automatic Parallelization of Mathematical Models**
  - Parallelism over the numeric solver method.
  - Parallelism over time.
  - **Parallelism over the model equation system**
    - ... with fine-grained task scheduling

• **Coarse-Grained Explicit Parallelization Using Components**
  - The programmer partitions the application into computational components using strongly-typed communication interfaces.
  - Co-Simulation, Transmission-Line Modeling (TLM)

• **Explicit Parallel Programming**
  - Providing general, easy-to-use explicit parallel programming constructs within the *algorithmic* part of the modeling language.
    - OpenCL, CUDA, ...
Gained speedup

- Intel Xeon E5520 CPU (16 cores) 26
- NVIDIA Fermi-Tesla M2050 GPU (448 cores) 115

Speedup comparison to sequential algorithm on Intel Xeon E5520 CPU
Peter Fritzson
Principles of Object Oriented Modeling and Simulation with Modelica 2.1


- OpenModelica
  - www.openmodelica.org

- Modelica Association
  - www.modelica.org
New Introductory Book
September 2011
232 pages

Wiley
IEEE Press

For Introductory Short Courses on Object Oriented Mathematical Modeling
Announcements, Coming Workshops

• Call for Presentations

• OpenModelica Workshop
  • Feb 6, 2012. www.openmodelica.org, Linköping, Sweden
    Applications and tool developments in the OpenModelica Open Source Effort.

• MODPROD Workshop on Model-Based Development
Summary

Multi-Domain Modeling

Visual Acausal Component Modeling

www.modelica.org – Language, Standard Library
www.openmodelica.org – Open Source Tool

Typed Declarative Textual Language

Thanks for listening!

Modelica® is a registered trademark of Modelica Association